Naval Research Laboratory

Washington, DC 20375-5320

NRL/FR/5591--08-10,176

Multi-Service Domain Protecting Interface Architecture

Christopher L. Robson

Center for Computational Science

Information Technology Division

December 19, 2008

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 19-12-2008	**2. REPORT TYPE** Formal Report	**3. DATES COVERED** *(From - To)*

4. TITLE AND SUBTITLE Multi-Service Domain Protecting Interface Architecture	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Christopher L. Robson	**5d. PROJECT NUMBER** 55-J955-A7
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory Washington, DC 20375-5320	**8. PERFORMING ORGANIZATION REPORT NUMBER** NRL/FR/5590--08-10,176
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Alex Lovett OUSD (AT&L) 3020 Defense Pentagon, Room 3D285 Washington, DC 20301-3020	**10. SPONSOR / MONITOR'S ACRONYM(S)**
	11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Protecting the confidentiality of data from unwanted disclosure or access is a common requirement for private and public networks. The integrity of the data is considered a key priority. A common technology used to assure the confidentiality, integrity, and availability of data is cryptography. Encrypting/decrypting the data is accomplished by using cryptographic technology. The data is protected from public disclosure by segmenting the data into private domains. This is accomplished through firewalling. Data availability is guaranteed by applying technology that assures the systems that host, transport, and protect the data are free from unwanted control. The technologies used to protect sensitive data are always undergoing revitalization to assure protection against continuing threats. This report proposes the Multi-Service Domain Protecting Interface Architecture (MSDPI), a new cost-reducing interface for encryption devices that protects networks and sensitive data.

15. SUBJECT TERMS

Multi-service domain protecting interface, encryption device, Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions, plain text domain, cypher text domain, Multi-Protocol Label Switch

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Christopher L. Robson
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	Unlimited	13	**19b. TELEPHONE NUMBER** *(include area code)* (202) 404-3138

CONTENTS

MULTI-SERVICE DOMAIN PROTECTING INTERFACE ARCHITECTURE

1. INTRODUCTION

The technology demonstrated in this report will provide many benefits to the government and private sector. It provides an effective way to maintain the technological readiness of encryption devices. It enhances the capabilities of existing information assurance techniques used today to protect the confidentiality and unwanted disclosure of sensitive data. It provides a deployable mechanism for policy control between protected domains while still assuring the policy is not compromised by non-peering domains. Further, this technology is not only beneficial to government organizations but can provide protection to such institutions as the banking industry and medical organizations for protecting private citizens' personal information. A key success to this design is its ability to use existing communication standards without any unique modifications typically found in today's encryption devices. This has the effect of increasing the flexibility of the system, allowing it to adapt to changes within an infrastructures. Because of the ease to modify and decouple components, the typical re-engineering and logistics costs associated with encryption device changes can be reduced or avoided altogether. This makes the design attractive to not only the U.S. Government, but to financial institutions such as Wall Street or medical organizations such as the typical hospital or HMO as well.

2. MULTI-SERVICE DOMAIN PROTECTING INTERFACE

The Multi-Service Domain Protecting Interface (MSDPI) defines a technology that functions as a secure interface between the network (Cypher Text Domain) and the encryption engine (Plain Text Domain). The MSDPI uses a configurable technique to determine the network structure and adapt to that technology at runtime. Simply put, MSDPI "wraps" the encryption engine with the network technology in use at the time. This wrapping function is transparent to both the encryption engine as well as the network. Further, it is envisioned that with the proper implementation, the MSDPI can adapt to multiple network technologies on the fly as they change.

The MSDPI architecture has two components that are used to achieve its goal. First MSDPI relies on Pseudo Wire Emulation (PWE) or similar technologies such as L2VPN. This provides the wrapping function of the data flow. The second function, and key to the MSDPI architecture, is the Initiator-Responder (I-R). I-R is used for all control plane signaling and data flow control through the encryption engine. I-R communications is accomplished using the Session Initiation Protocol (SIP) for Instant Messaging and Presence Leveraging Extensions (SIMPLE). Further, I-R exploits SIP standard-based signaling, SIP authentication, SIP policy and validation, securing communications between the I-Rs. By using the I-R design, the encryption engine is protected from compromise, disclosure and denial of service.

Manuscript approved August 11, 2008.

3. DESIGN

3.1 Overview

Figure 1 illustrates the functional architecture of this encryptor. This design exploits two prior technologies detailed in Refs. 1 and 2. The MSDPI interface may exist within a single unit or consist of many integrated units.

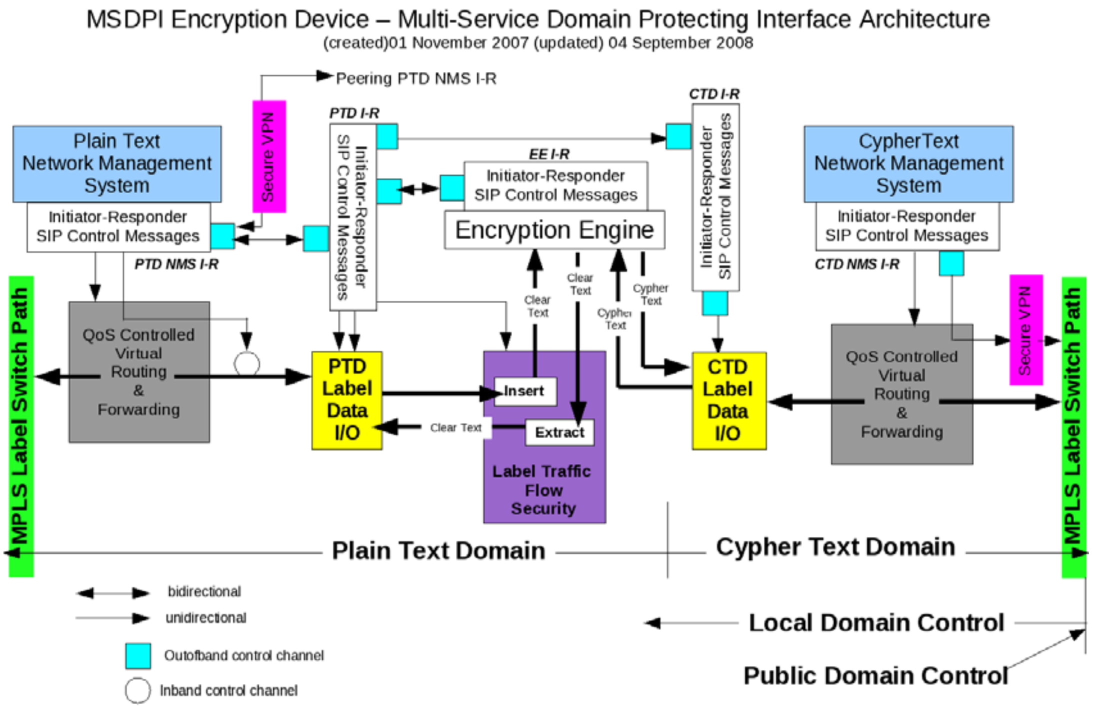

MSDPI Encryption Device – Multi-Service Domain Protecting Interface Architecture
(created)01 November 2007 (updated) 04 September 2008
Peering PTD NMS I-R
Plain Text Network Management System
Initiator-Responder SIP Control Messages
PTD NMS I-R
Secure VPN
PTD I-R
Initiator-Responder SIP Control Messages
EE I-R
Initiator-Responder SIP Control Messages
Encryption Engine
CTD I-R
Initiator-Responder SIP Control Messages
CypherText Network Management System
Initiator-Responder SIP Control Messages
CTD NMS I-R
Secure VPN
MPLS Label Switch Path
QoS Controlled Virtual Routing & Forwarding
PTD Label Data I/O
Clear Text
Clear Text
Cypher Text
Cypher Text
Insert
Extract
Clear Text
Label Traffic Flow Security
CTD Label Data I/O
QoS Controlled Virtual Routing & Forwarding
MPLS Label Switch Path
Plain Text Domain
Cypher Text Domain
bidirectional
unidirectional
Outofband control channel
Inband control channel
Local Domain Control
Public Domain Control

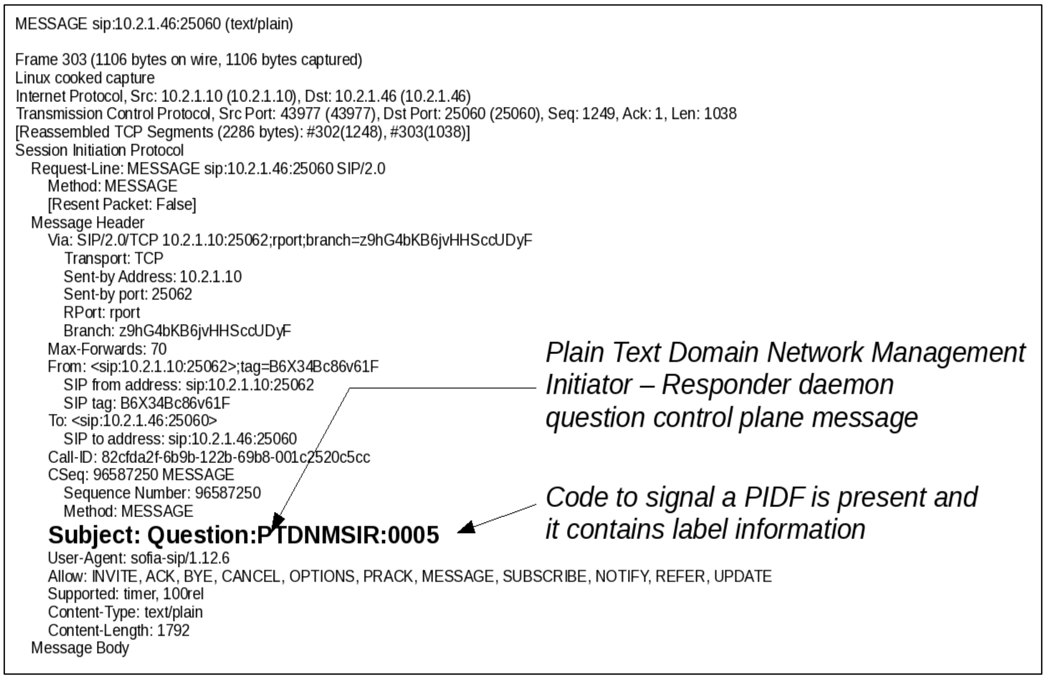
Fig. 1 – MSDPI Initiator-Responder architecture

3.2 Network Interface

A feature of this encryptor architecture is the ability to interface to any network infrastructure type, either locally administered or service provider. The ability to interface to multinetwork types is possible because this encryptor exploits the “Sandwich” technology detailed in Ref. 2.

3.3 SIP MESSAGE Dialog

The SIP dialog used between the I-R processes is exactly as defined in Ref. 1. The MESSAGE request requires the subject to be a question or an answer. The format of the question subject must begin with the key word “Question” and the answer subject line must begin with the key word “Answer.” Each of these subject line key words must be followed by a message type and code such that a subject line format is: Question:[type]:[code] or Answer:[type]:[code]. Each of the keys words must be separated by a “:”. Spaces are ignored. Figure 2 illustrates a typical SIP MESSAGE dialog.



```
MESSAGE sip:10.2.1.46:25060 (text/plain)

Frame 303 (1106 bytes on wire, 1106 bytes captured)
Linux cooked capture
Internet Protocol, Src: 10.2.1.10 (10.2.1.10), Dst: 10.2.1.46 (10.2.1.46)
Transmission Control Protocol, Src Port: 43977 (43977), Dst Port: 25060 (25060), Seq: 1249, Ack: 1, Len: 1038
[Reassembled TCP Segments (2286 bytes): #302(1248), #303(1038)]
Session Initiation Protocol
   Request-Line: MESSAGE sip:10.2.1.46:25060 SIP/2.0
      Method: MESSAGE
      [Resent Packet: False]
   Message Header
      Via: SIP/2.0/TCP 10.2.1.10:25062;rport;branch=z9hG4bKB6jvHHSccUDyF
         Transport: TCP
         Sent-by Address: 10.2.1.10
         Sent-by port: 25062
         RPort: rport
         Branch: z9hG4bKB6jvHHSccUDyF
      Max-Forwards: 70
      From: <sip:10.2.1.10:25062>;tag=B6X34Bc86v61F
         SIP from address: sip:10.2.1.10:25062
         SIP tag: B6X34Bc86v61F
      To: <sip:10.2.1.46:25060>
         SIP to address: sip:10.2.1.46:25060
      Call-ID: 82cfda2f-6b9b-122b-69b8-001c2520c5cc
      CSeq: 96587250 MESSAGE
         Sequence Number: 96587250
         Method: MESSAGE
      Subject: Question:PTDNMSIR:0005
      User-Agent: sofia-sip/1.12.6
      Allow: INVITE, ACK, BYE, CANCEL, OPTIONS, PRACK, MESSAGE, SUBSCRIBE, NOTIFY, REFER, UPDATE
      Supported: timer, 100rel
      Content-Type: text/plain
      Content-Length: 1792
   Message Body
```

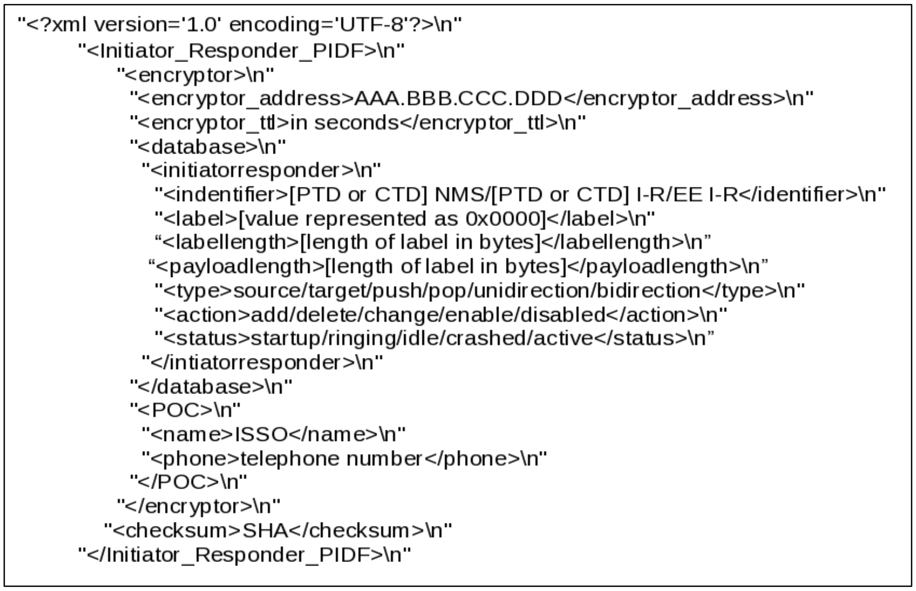
Plain Text Domain Network Management Initiator – Responder daemon question control plane message

Code to signal a PIDF is present and it contains label information

Fig. 2 – Example of an I-R MESSAGE dialog message

3.4 PIDF Format

The generic format of the PIDF used within this architecture is illustrated in Fig. 3. Table 1 provides the default tags found in the MESSAGE request and the functions the tags perform.



```
"<?xml version='1.0' encoding='UTF-8'?>\n"
     "<Initiator_Responder_PIDF>\n"
        "<encryptor>\n"
         "<encryptor_address>AAA.BBB.CCC.DDD</encryptor_address>\n"
         "<encryptor_ttl>in seconds</encryptor_ttl>\n"
         "<database>\n"
          "<initiatorresponder>\n"
           "<indentifier>[PTD or CTD] NMS/[PTD or CTD] I-R/EE I-R</identifier>\n"
           "<label>[value represented as 0x0000]</label>\n"
           “<labellength>[length of label in bytes]</labellength>\n”
          “<payloadlength>[length of label in bytes]</payloadlength>\n”
           "<type>source/target/push/pop/unidirection/bidirection</type>\n"
           "<action>add/delete/change/enable/disabled</action>\n"
           "<status>startup/ringing/idle/crashed/active</status>\n”
          "</intiatorresponder>\n"
         "</database>\n"
         "<POC>\n"
          "<name>ISSO</name>\n"
          "<phone>telephone number</phone>\n"
         "</POC>\n"
        "</encryptor>\n"
       "<checksum>SHA</checksum>\n"
     "</Initiator_Responder_PIDF>\n"
```

Fig. 3 – Basic I-R PIDF control message

Table 1 – PIDF Tag Definitions

TAG	FUNCTION
encryptor	Target encryptor address
encryptor_ttl	PIDF time to live, when expired PIDF is ignored
database	New database follows
initiatorresponder	New Initiator – Responder control plane section
identifier	Identifies the specific Initiator - Responder
label	Label identifier, signals a bounded traffic flow to the EE
labellength	Length of the label.
payloadlength	Length of the payload; this is required by the EE.
type	Identifies the label as source, destination, unidirectional, bidirectional
action	How the Initiator – Responder is to process labels
status	Initiator – Responder status
POC name/phone	ISSO contact information
checksum	Checksum or user authentication mechanism

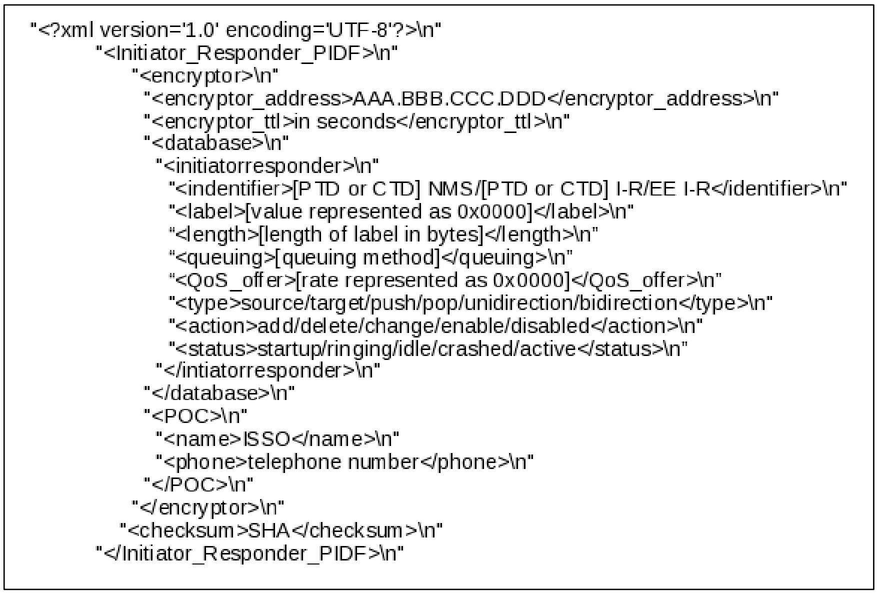
4. ARCHITECTURE

4.1 Definitions of Components

To help clarify the discussion, let's begin by defining some additional MSDPI component names shown in Fig. 1. The Initiator-Responder (I-R) used to control message traffic within the local domain control is called the Plan Text Domain (PTD) Network Management System (NMS) and will be referred to as the PTD NMS I-R. The I-R that controls the local domain side Label Data I/O will be referred to as the PTD I-R. The I-R controlling the encryption engine (EE) is the EE I-R. The CTD Label Data I/O I-R is called the CTD I-R. CTD traffic management and flow is managed and controlled by the CTD NMS I-R. Each of these components is further defined in Section 4.4.

4.2 Control Messages

As previously discussed, all control message traffic between the I-Rs will employ the technology first detailed in Ref. 1. Figure 4 is an example of a possible PTD NMS I-R SIP SIMPLE MESSAGE PIDF.



```
"<?xml version='1.0' encoding='UTF-8'?>\n"
          "<Initiator_Responder_PIDF>\n"
                "<encryptor>\n"
                  "<encryptor_address>AAA.BBB.CCC.DDD</encryptor_address>\n"
                  "<encryptor_ttl>in seconds</encryptor_ttl>\n"
                  "<database>\n"
                    "<initiatorresponder>\n"
                      "<indentifier>[PTD or CTD] NMS/[PTD or CTD] I-R/EE I-R</identifier>\n"
                      "<label>[value represented as 0x0000]</label>\n"
                      “<length>[length of label in bytes]</length>\n"
                      “<queuing>[queuing method]</queuing>\n"
                      “<QoS_offer>[rate represented as 0x0000]</QoS_offer>\n"
                      "<type>source/target/push/pop/unidirection/bidirection</type>\n"
                      "<action>add/delete/change/enable/disabled</action>\n"
                      "<status>startup/ringing/idle/crashed/active</status>\n"
                    "</intiatorresponder>\n"
                  "</database>\n"
                  "<POC>\n"
                    "<name>ISSO</name>\n"
                    "<phone>telephone number</phone>\n"
                  "</POC>\n"
                "</encryptor>\n"
              "<checksum>SHA</checksum>\n"
          "</Initiator_Responder_PIDF>\n"
```

Fig. 4 – Example of PTD NMS I-R control message

4.3 Managing Policy between MSDPI Interfaces

The MSDPI adheres to the concepts developed in Ref. 1 for managing policy. All PIDF exchanges remain within the LDC and peering MSDPIs. Each LDC PTD NMS I-R is responsible for negotiating policy between peering MSDPIs. Figure 5 illustrates this negotiation sequence. Figure 6 illustrates the architecture of peering MSDPI policy dialog. The assurance of policy between the PTD and CTD is by mapped (configured) assignment.

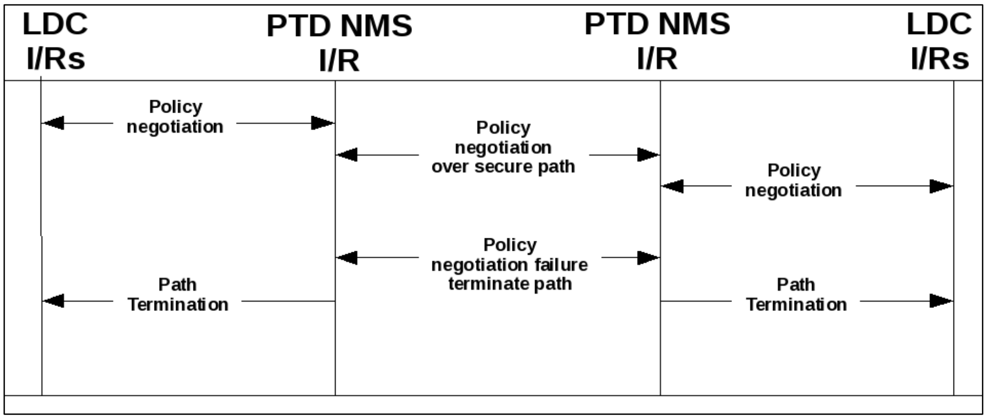
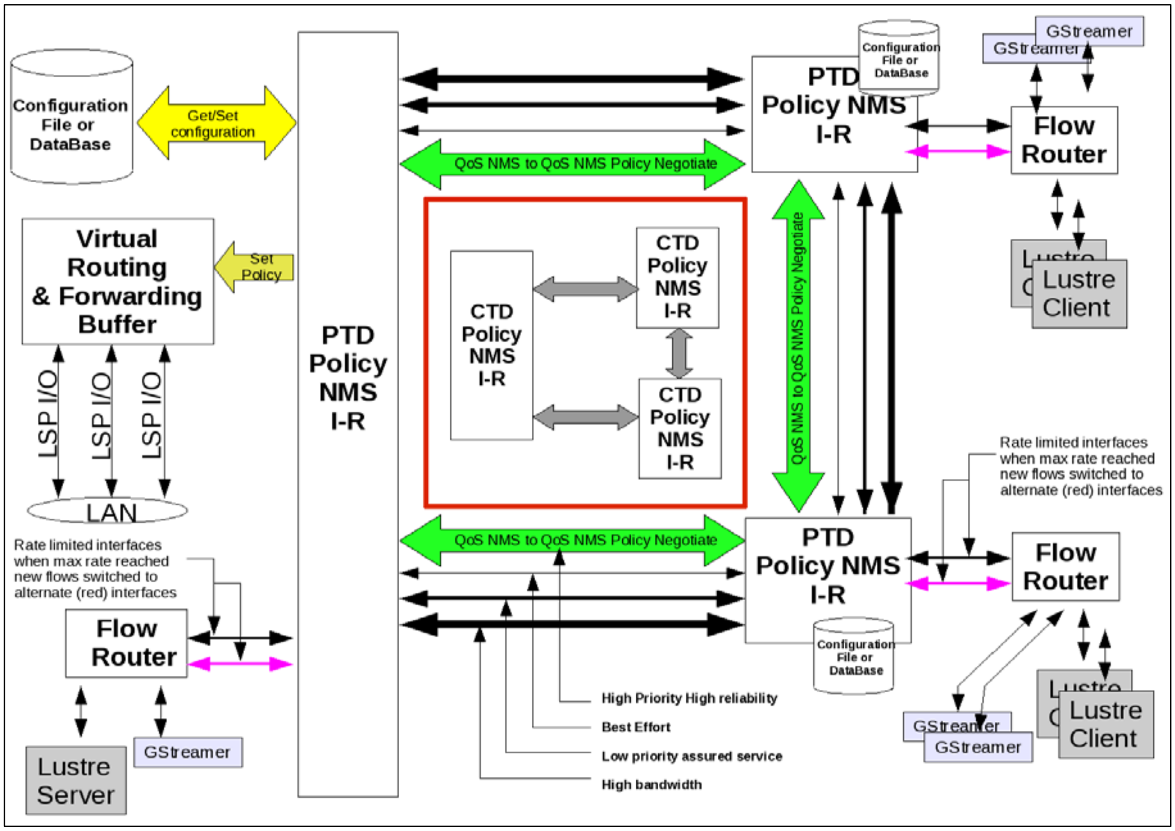


Fig. 5 – PTD NMS I-R to PTD NMS I-R policy negotiation


Configuration File or DataBase
Get/Set configuration
Virtual Routing & Forwarding Buffer
Set Policy
LSP I/O
LSP I/O
LSP I/O
LAN
PTD Policy NMS I-R
QoS NMS to QoS NMS Policy Negotiate
CTD Policy NMS I-R
CTD Policy NMS I-R
CTD Policy NMS I-R
QoS NMS to QoS NMS Policy Negotiate
PTD Policy NMS I-R
Configuration File or DataBase
GStreamer
GStreamer
Flow Router
Lustre Client
Lustre Client
Rate limited interfaces when max rate reached new flows switched to alternate (red) interfaces
QoS NMS to QoS NMS Policy Negotiate
PTD Policy NMS I-R
Configuration File or DataBase
Flow Router
GStreamer
GStreamer
Lustre Client
Lustre Client
Rate limited interfaces when max rate reached new flows switched to alternate (red) interfaces
Flow Router
Lustre Server
GStreamer
High Priority High reliability
Best Effort
Low priority assured service
High bandwidth

Fig, 6 – Example of peering LDC PTD NMS I-R policy flow

4.4 Component Description

This section discusses each of the MSDPI I-R daemons, describing each daemon and the functions it performs.

4.4.1 LDC PTD Network Management System Initiator-Responder

The LDC PTD Network Management System Initiator-Responder (PTD NMS I-R) is possibly the most important component of this architecture. This is because it controls the actions of the Virtual Routing and Forwarding Buffers, the PTD I-R and indirectly through the PTD I-R, the PTD Label Data I/O, the Encryption Engine (EE) I-R, the EE, the CTD I-R, and the CTD Label Data I/O. Further the LDC PTD NMS I-R will negotiate policy with peering LDC PTD NMS I-Rs. Figure 1 illustrates the connection architecture between each of the I-R daemons showing how the LDC PTD NMS I-R, as the master daemon, controls all other I-R daemons within the LDC. Figure 7 illustrates the policy negotiation sequence taken place within the MSDPI. Figure 8 illustrates how the PTD NMS I-R controls the other I-R and processing units (Label Data I/O or EE) either directly or by issuing and responding to negotiation requests from the other component controlling I-Rs.

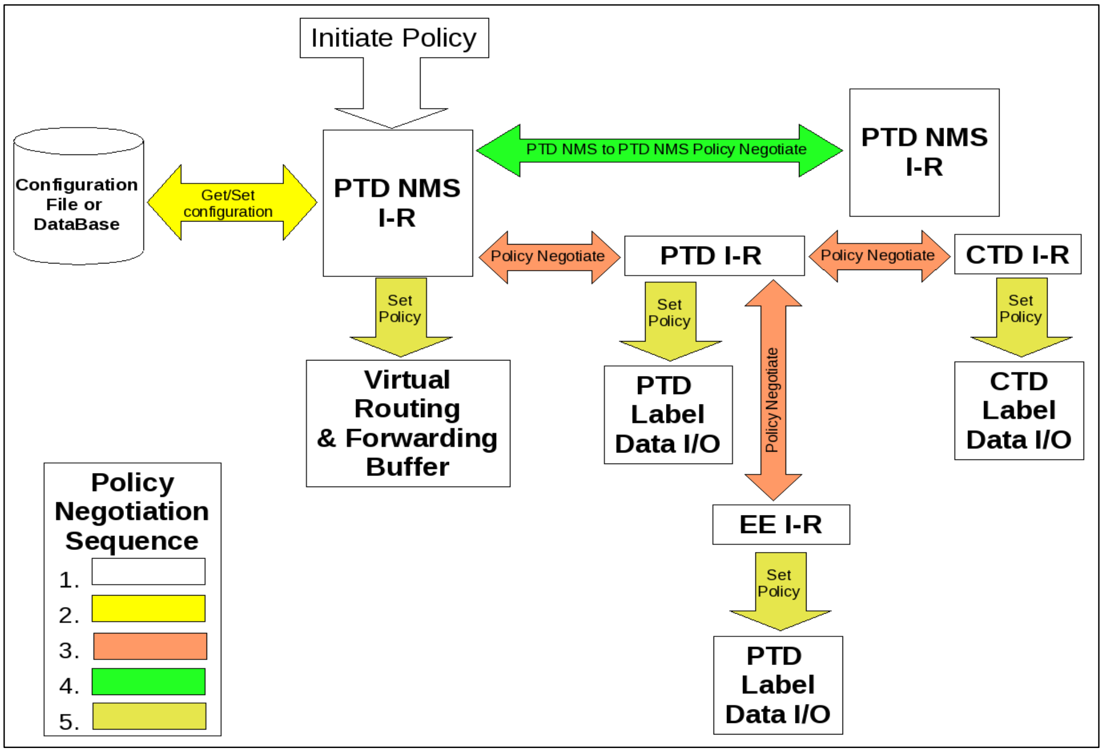


Fig. 7 – I-R policy negotiation sequence

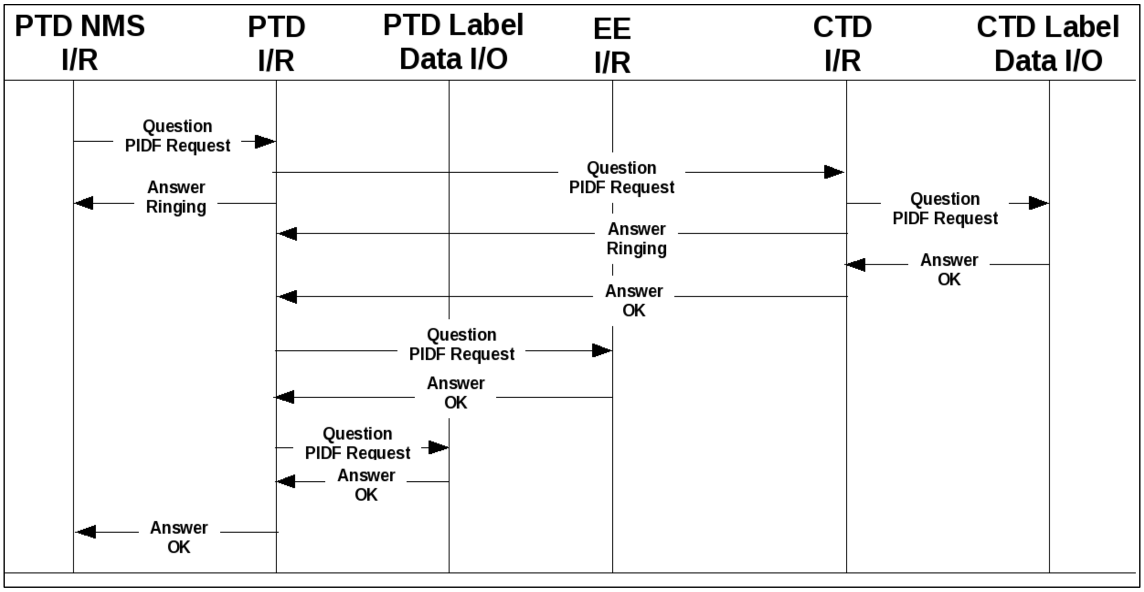


Fig. 8 – LDC I-R command flow

4.4.2 LDC PTD Initiator-Responder

The LDC PTD Initiator-Responder (PTD I-R) controls the actions of the LDC PTD Label Data I/O and the LDC CTD I-R. Figure 9 illustrates the sequence of change policy events.

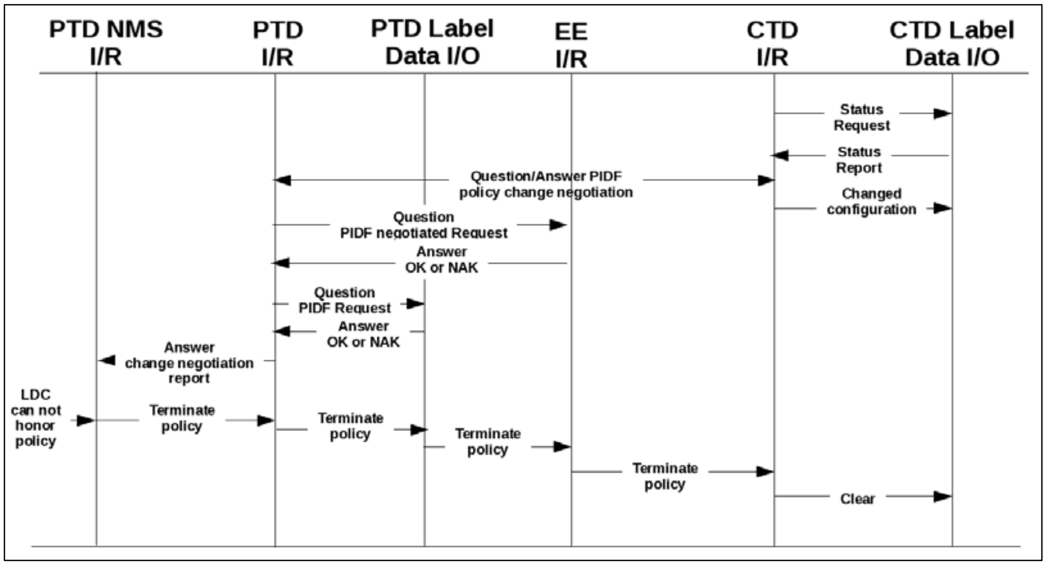


Fig. 9 – LDC I-R policy status/negotiation

4.4.3 Encryption Engine Initiator-Responder

The Encryption Engine Initiator-Responder (EE I-R) controls the actions of the encryption engine.

4.4.4 CTD Initiator-Responder

The CTD Initiator-Responder (CTD I-R) controls the action of the CTD Label Data I/O. To improve security, the CTD I-R has no direct interface to the Cypher Text Domain (PDC), it will not accept any control signaling from the CTD and is only accessible from the PTD and is controlled by the PTD I-R. Figure 1 illustrates this showing the out-of-band control command arrow going from the PTD I-R to the CTD I-R.

4.4.5 CTD Network Management System Initiator-Responder

The CTD Network Management System Initiator-Responder (CTD NMS I-R) controls the actions of the CTD Virtual Routing and Forwarding Buffer (VRFB). To provide IA, it is only accessible locally or through an existing SA between managed peering CTD NMS I-Rs.

5. CONTROL PLANE

The MSDPI binds I-O flows to a “label.” The MSDPI binds the ending of a flow by extracting out the length field of the original header. To assure peering MSDPIs are synchronized, the MSDPI assigned as the session Initiator would be required to inform a peering Responder about any labels. Figure 10 illustrates the point where the MPLS header is designated as the MSDPI label, thus establishing the beginning of a flow and the point in which the EE is to begin encrypting or decrypting the traffic data.

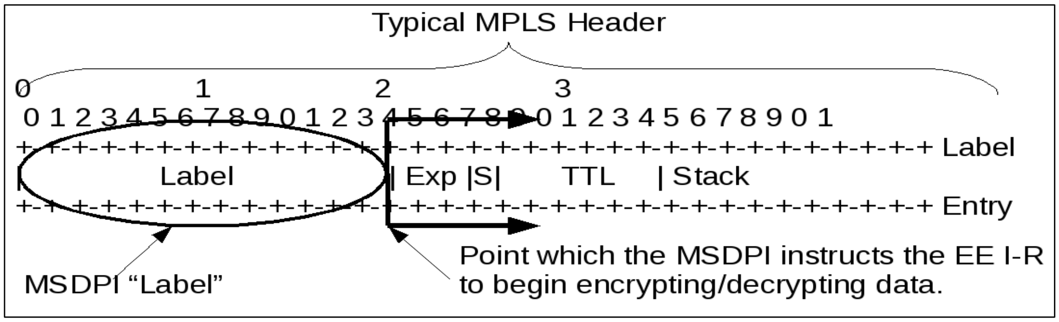


Fig. 10 – MSDPI label encryption

6. DATA PLANE

6.1 PTD/CTD MSDPI Virtual Routing and Forwarding Buffers

The MSDPI control of data traffic flows consist of two basic functions in both the PTD and CTD, policy control and the data traffic. To control PTD traffic flow, the MSDPI uses the concept of a Virtual Routing and Forwarding Buffer. The MSDPI VRFBs function similar to how L3VPN service uses VRF by associating policy to each VRFB to control traffic flows.

6.2 PTD/CTD Label Data I/O

The PTD and CTD Data I/O is the interface buffer that accepts and transmits data traffic through the encryption engine and provides congestion control.

7. INITIATOR-RESPONDER PROTOTYPE ARCHITECTURE

The prototype developed for this effort consists of three basic components: the Initiator, the Responder, and the Functional Module. Figure 11 illustrates the relationship between each of the prototype subsystems. For example, the Responder listens on a specified port for incoming SIP SIMPLE MESSAGE traffic. When it receives a SIMPLE MESSAGE, it will decode the message subject line and call the appropriate Functional Module based on the "type" code, passing any attached PIDF data to the Functional Module. Upon receiving a request from the Responder to begin processing a SIMPLE MESSAGE, the Functional Module will first decode the command "code" subject line key word so it understands how to process any attached PIDF data. If the SIP subject contains the "Question" key word, when the Functional Module has completed processing the incoming request, it will build a response SIP subject line containing the "Answer" key word and any appropriate "type" and command "code" key word values and pass the "Answer" responses to the Responder. The Initiator is typically only run at system boot-up to initiate any start-up dialog between MSDPIs.

A prototype has been developed as a guide to the implementer of the MSDPI architecture. The prototype was first developed for Ref. 1. However, for this effort, it was enhanced to support a distributed architecture. For example, signaling between subsystems is accomplished through IOCTL.

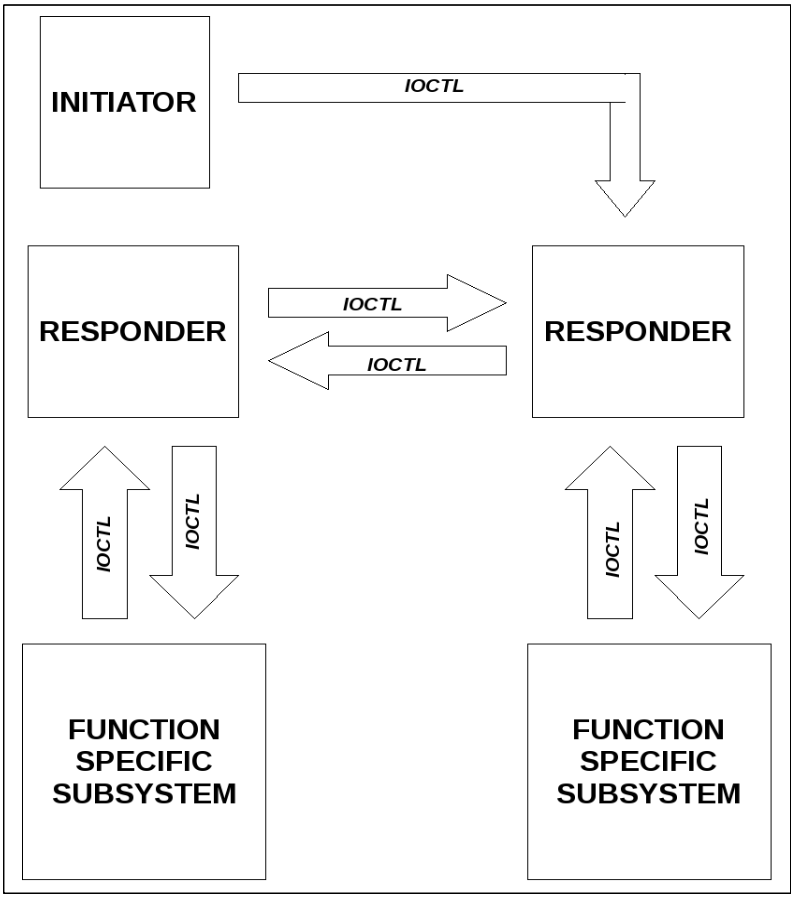


Fig. 11 – I-R prototype architecture

8. CONCLUSION

It is believed the technology demonstrated in this report will provide many benefits to the government and private sector. It provides an effective way to maintain the technological readiness of encryption device technology. It enhances the capabilities of existing information assurance techniques used today to protect the confidentiality and unwanted disclosure of sensitive data. It provides a deployable mechanism for policy control between protected domains while still assuring the policy is not compromised by non-peering domains. Further, this technology is not only beneficial to government organizations but can provide protection to such institutions as the banking industry and medical organizations for protecting private citizens' personal information.

REFERENCES

1. C. Robson, "Session Initiation Protocol Network Encryption Device Plain Text Domain Discovery Service," NRL/FR/5591--07-10,156, pp. 1-16, December 7, 2007.

2. C. Robson, "How to Use FASTLANEs to Protect IP Networks," NRL/MR/5590--06-8979, pp. 1-23, August 18, 2006.